2023012838 中美 张鑫烁

No.

Date

第一章.

一. 名词解释

1. 操作系统是配置在计算机硬件上的第一层软件，是对硬件系统的首次扩充，其主要作用是管理硬件设备，提高它们的利用率和系统吞吐量，并为用户和应用程序提供一个简单的接口，以便于用户和应用程序使用硬件设备.

2. 实时系统是指系统能及时响应外部事件的请求，在规定时间内完成对该事件的处理，并控制所有实时任务协调一致地运行

3. 当进程A要访问某资源时，必须先提出请求. 若此时该资源空闲，系统便可将其分配给进程A使用. 此后若再有其他进程也要访问该资源，则必要进程A未用完. 其他进程均必须等待. 仅当进程A访问完并释放后，才允许另一进程对该资源进行访问. 称为互斥共享

三. 填空题.

1. 硬件系统、软件系统

3. 不可抢夺资源、可抢夺资源

5. 共享性、互斥性

第二章.

一. 名词解释

3. 临界区是指在多线程或多进程环境中，访问共享资源的一段代码区域，当多个线程或进程需要访问同一共享资源时，为了避免数据不一致或其他错误，需要确保在同一时刻只有一个线程或进程能够进入临界区访问共享资源，其他线程或进程必须等待

4. 进程同步是指多个进程在执行过程中，通过协调它们的执行顺序和时机，以确保对共享资源的正确访问和使用，避免出现数据不一致等问题.

Date

三. 填空题.

1. 动态性，异步性，结构性

5. 动态. 静态

6. 进程同步

8. 共享存储器通信；消息传递通信；管道通信

9. 就绪；运行；阻塞

五. 问答题

2. 前驱关系：S_1是S_2. S_3的前驱；S_2是S_4. S_5的前驱；S_3是S_6的前驱；
S_4. S_5. S_6是S_7的前驱.

信号量机制设计：① 定义信号量 ~~semaphore S2=0; S3=0; S4=0;~~ S5=0

semaphore S2=0, S3=0, S4=0, S5=0, S6=0, S7=0;

② 进程代码：

```
void S1(){ V(S2);
           V(S3); }

void S2(){ P(S2);
           V(S4);
           V(S5); }

void S3(){ P(S3);
           V(S6);}

void S4(){ P(S4);
           V(S7);}

void S5(){ P(S5);
           V(S7);}

void S6(){ P(S6);
           V(S7);}

void S7(){ P(S7);
           P(S7);}
```

3. ① 定义信号量：

```
semaphore mutex=1, apple=0,
          orange=0;
```

② 进程代码：

```
// 父亲进程
void father(){
  while(1){
    P(mutex);
    if(放入的是苹果){
      V(apple);
    }else{
      V(orange);
    }
    V(mutex);}}

// 儿子进程
void son(){
  while(1){
    P(orange);
    P(mutex);
    V(mutex);
  }}

// 女儿进程
void daughter()
{ while(1){
    P(apple);
    P(mutex);
    V(mutex);
  }}
```

Campus

2023012838 中美 张翁光荣

No.

Date

⑤ HRRN

完成时间：A(3)·B(9)·C(13) E(15) D(20)

周转：A(3-0=3)·B(9-2=7)·C(13-4=9) E(15-8=7) D(20-6=14)

带权周转：A(3÷3=1) B(7÷6≈1.17)·C(9÷4=2.25)

E(7÷2=3.5)·D(14÷5=2.8)

平均周转：(3+7+9+7+14)÷5=8

平均带权：2.144

⑥ RR

完成：A(5)·B(17)·C(9)·D(20)·E(11)

周转：A(5-0=5) B(17-2=15)·C(9-4=5)·D(20-6=14) E(11-8=3)

带权周转 A(5÷3≈1.67) B(15÷6=2.5) C(5÷4=1.25)

D(14÷5=2.8) E(3÷2=1.5)

平均周转：8.4 平均带权：1.944

2. 任务A：0ms时，松弛度=20-0-10=10

20ms时，松弛度=40-20-10=10 以此类推

B： 0ms时，松弛度=40，50ms时，松弛度=30 以此类推

C：0ms时，松弛度=35，50ms时，松弛度=35

调整顺序：在0ms时，A松弛度10最小，先执行A；

执行完A后，不检查B和C，C松弛度35小于B

为40，执行C；之后才执行B，按照此方式，依据

每个时刻各任务松弛度大小，优先执行松弛度

最小的任务来调整

No.

Date . .

3. (1) 分配矩阵

P0 = 0044 - 0012 = 0032

P1 = 2750 - 1750 = 1000

P2 = 36 10 10 - 2356 = 1354

P3 = 0984 - 0652 = 0332

P4 = 066 10 - 0656 = 0014

安全性检查

Work = Available = 1622, P

从进程找 Need ≤ Work 的, P0 满足,

Work = 1622 + 0032 = 1654

P3 满足, Work = 1654 + 0332 = 1986

P4 满足, Work = 1986 + 0014 = 199 10

P1 满足, Work = 199 10 + 1000 = 299 10

P2 满足, 所以系统处于安全状态

(2) P2 请求 Request (1, 2, 2, 2)

Available = 1622 - 1222 = 0400

分配后 P2 的 Need 变为 2356 - 1222 = 1134

Work = 0400. 未找不到 Need ≤ Work 的进程,

所以不能将资源分配给它

(3) 因为按(1)中分配后系统不安全, 但并不意味着

它即进入到死锁状态, 只是有在死锁风险, 若后续进程

资源请求和释放情况合理, 仍可能避免死锁,

所以系统不会立即进入死锁状态。

No.

Date

第三章. 处理机调度和死锁

一. 名词解释

2. 处理机调：根据一定的调度算法，从就绪队列中选择一个进程分配CPU资源，使其能多在处理机上运行的过程，目的是提高系统资源利用率和系统性能。

3. 周转时间：从作业提交给系统开始，到作业完成为止的时间间隔。它反映了作业在系统中停留的总时长，包括作业等待时间和执行时间，是衡量系统调度性能的重要指标之一。

4. 死锁：在多进程并发环境下，多个进程因竞争资源而造成的一种僵局状态。这些进程都在等待其他进程释放其所占资源，导致它们都无法继续推进，若无外力干预，将永远处于阻塞状态

三. 填空题.

2. 提交. 后备. 完成

6. 预防死锁. 避免死锁. 检测死锁. 解除死锁

五. 问答题

1. ① FCFS

完成时间：A(3). B(9). C(13). D(18). E(20)

周转时间：A(3). B(9-2=7). C(13-4=9). D(18-6=12). E(20-8=12)

带权周转时间：A(3÷3=1). B(7÷6≈1.17). C(9÷4=2.25)

D(12÷5=2.4). E(12÷2=6)

平均周转时间：(3+7+9+12+12)÷5=8.6

平均带权周转时间：(1+1.17+2.25+2.4+6)÷5=2.564

② SJF

完成：A(3) C(7) E(9) D(14) B(20)

周转 A(3-0=3) B(20-2=18) C(7-4=3). D(14-6=8) E(9-8=1)

带权周转：A(3÷3=1). B(18÷6=3). C(3÷4=0.75) D(8÷5=1.6)

E(1÷2=0.5)

平均周转：6.6 带权平均：1.3

2023012838 中关 张鑫烁

No.

Date

4. ① 定义信号量：

semaphore empty1=1, full1=0, empty2=1, full2=0;

② 进程代码：

// 进程PA

void PA(){

while(1){

P(empty1);

V(full1);}

// 进程PB

void PB(){

while(1){

P(full1);

P(empty2);

V(full2);

V(empty1);}

// 进程PC

void PC(){

while(1){

P(full2);

V(empty2);}

5. ① 定义信号量：

Semaphore chair=N, barber=0, customer=0, payment=0;

② 进程代码：

// 顾客进程

void customer(){

P(chair);

V(customer);

P(barber);

V(payment);

P(payment);

V(chair);}

// 理发师进程

void barber(){

while(1){

P(customer);

V(barber);

P(payment);

V(payment);

}

}